

# IA Proyecto 2: Clustering

Jorge Luis Neira Riveros  
*Ciencia de la Computación*  
*Univ. de Ingeniería y Tecnología*  
Lima, Perú  
jorge.neira@utec.edu.pe

José Adrián Porres Brugué  
*Ciencia de la Computación*  
*Univ. de Ingeniería y Tecnología*  
Lima, Perú  
jose.porres@utec.edu.pe

Efraín Córdova  
*Ciencia de la Computación*  
*Univ. de Ingeniería y Tecnología*  
Lima, Perú  
efrain.cordova@utec.edu.pe

**Abstract**—En este trabajo, presentamos nuestra experimentación sobre tres algoritmos de *clustering* de inteligencia artificial: "Kmeans", "DBScan", y "GMM". Implementamos los algoritmos para aplicar *clustering* y segmentación de tumores sobre una serie de imágenes de cerebros humanos. Además, realizamos consultas sobre nuevas imágenes de tumores y encontramos los más similares.

**Index Terms**—IA, Kmeans, DBScan, GMM, Clustering

## I. INTRODUCCIÓN

En el presente proyecto, implementamos tres métodos de *clustering* utilizados para segmentar. Estos modelos son no supervisados, es decir, los datos no presentan etiqueta alguna, por lo que la determinación de los *clusters* obtenidos depende de nosotros mismos. Estos algoritmos realizan una agrupación de los datos, generando etiquetas a cada grupo. Aplicaremos estos algoritmos a las imágenes de una base de datos de tumores en el cerebro humano. Obtendremos los vectores característicos del grupo correspondiente al tumor y normalizaremos sus dimensiones con el fin de aliviar las consultas futuras, para lo cual indexaremos en un *BallTree*. De las imágenes que no se utilizaron en la aplicación de los algoritmos, obtendremos sus vectores característicos y obtendremos las imágenes con tumores más similares.

## II. MODELOS

### A. Kmeans

Es un algoritmo de clasificación no supervisado que genera  $K$  grupos basados en sus características. El algoritmo se puede representar en los siguientes pasos:

- Seleccionar  $K$  datos, que representarán los  $K$  centroides de cada grupo.
- Clasificamos los datos de acuerdo al centroide más cercano.
- Actualizamos cada centroide con la media de los datos en su grupo.
- Volvemos a clasificar los datos y repetimos el proceso hasta que los centroides ya no se actualicen.

Este algoritmo puede mejorarse si escogemos los centroides iniciales de una mejor manera. Esta variación se llama *Kmeans++*, consiste en escoger el primer centroide más alejado de todos (en nuestro caso, escogimos el más cercano al origen) y los siguientes centroides son escogidos de tal forma que la distancia hacia los centroides ya seleccionados

sea la mayor posible. Es decir, escogemos un centroide 0 y luego el centroide 1 será el dato más alejado al centroide 0, luego de ello, el centroide 2 será el más alejado al 0 y al 1; así sucesivamente. La imagen 1 representa el proceso del algoritmo *Kmeans*

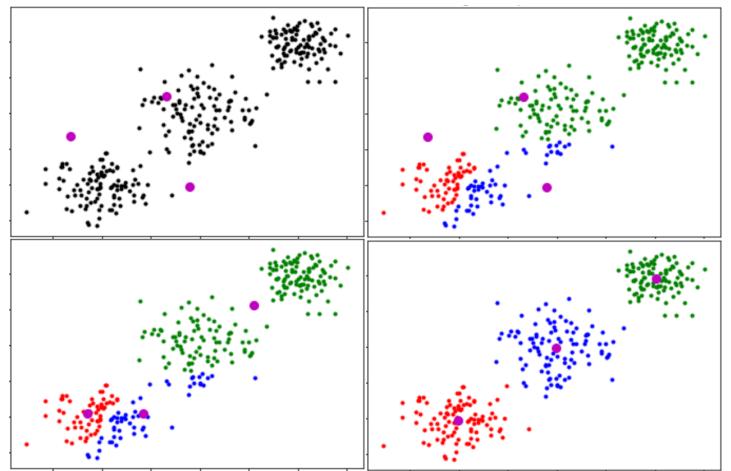


Fig. 1. Ejemplo del algoritmo Kmeans

### B. DBScan

Es un algoritmo no supervisado de *clustering* basado en la densidad espacial de los datos. Se define como hiperparámetros a un *threshold* y un radio de búsqueda. Luego, aplicamos un algoritmo similar a un *DFS*, donde la búsqueda de un *cluster* continuará, siempre y cuando, en el radio definido por el hiperparámetro, se encuentren por lo menos "*threshold*" vecinos. Se realizará este proceso para cada dato no visitado, de forma que todos sean clasificados. Si dos recorridos se cruzan, ambos deben combinarse en el mismo *cluster*. Los datos no etiquetados se consideran ruido. El algoritmo puede describirse en los siguientes pasos:

- Escoger un dato que no pertenezca a un *cluster* ni sea considerado ruido.
- Realizar una consulta de rango (usaremos un *KDTree*). De acuerdo al tamaño de los datos recogidos, considerarlo ruido o etiquetarlo en un nuevo *cluster*.
- Luego, repetimos este proceso con todos los datos que fueron encontrados, los cuales pertenecerán al mismo

*cluster*.

- Debemos repetir estos pasos hasta que cada nodo haya sido incluido en un *cluster* o sea considerado como ruido.

A diferencia del *Kmeans*, no sabemos cuántos *clusters* encontrará este algoritmo. La imagen 2 muestra cómo avanza el algoritmo dato a dato.

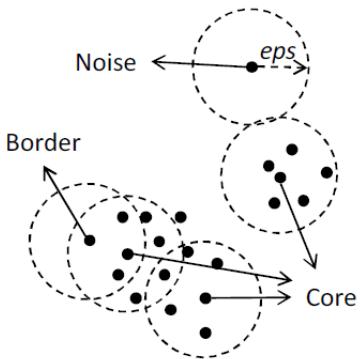


Fig. 2. Ejemplo del algoritmo DBScan

### C. Gaussian Mixture Models - GMM

Es un algoritmo no supervisado de *clustering* que utiliza  $K$  distribuciones probabilísticas sobre el conjunto de datos. Se puede definir como la generalización del *Kmeans*, donde, en lugar de definir que un objeto pertenece a una sola clase, en *GMM* cada objeto tiene cierta probabilidad de pertenecer a cualquier clase. En este modelo, el hiperparámetro  $K$  define cuántas campanas de Gauss utilizaremos sobre nuestros datos, y el hiperparámetro *iteraciones* define cuántas veces se realizará el algoritmo. Como sabemos, cada campana de Gauss está definida por una media y una covarianza, las cuales se irán moviendo hasta que se ajusten lo mejor posible a los datos. El algoritmo se puede definir así:

- Se inicializan  $K$  medias,  $K$  covarianzas aleatoriamente y los coeficientes de mezcla ( $1/K$  cada uno).
- Calculamos las probabilidades de cada objeto de pertenecer a cada grupo.
- Recalculamos los parámetros de medias, covarianzas y parámetros de mezcla.
- Calculamos el *likelihood*.
- Volvemos a ejecutar el algoritmo una cantidad de iteraciones o hasta que el *likelihood* tenga una tendencia a converger.

La imagen 3 muestra cómo se mueven las campanas de Gauss sobre los datos.

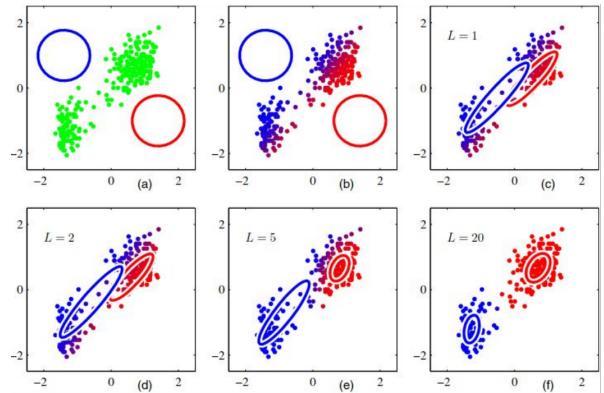


Fig. 3. Ejemplo del algoritmo GMM

## III. EXPERIMENTACIÓN

Aplicamos los algoritmos sobre las primeras 270 imágenes, de un total de 302, de la base de datos de tumores en cerebros humanos. Identificamos el *cluster* que representa al tumor y guardamos sus píxeles en un archivo. Cabe resaltar que los gráficos que muestran todas las clases en una imagen tienen clases con el mismo color, esto debido a que usamos una cantidad de colores base menor al de clases obtenidas.

### A. Consideraciones

- Las imágenes se ajustan a un tamaño de  $50 \times 50$ .
- Se consideran como datos no solo el valor del color de cada pixel, sino también su posición  $x$  y  $y$ , esto debido a que si solo consideramos el color, el color del tumor podría ser igual a otro lado del cerebro que no tiene tumor. Añadiendo la posición del pixel, aseguramos que la agrupación considere además la cercanía de los datos en la imagen.
- Para *Kmeans*, se utilizará un  $K$  igual a 25, ya que, como consideramos la cercanía de los datos, se necesitan más *clusters* para segmentar el tumor.
- Para *DBScan*, se utilizaron un radio entre 20 y 40, y un *threshold* igual a 10. Estos hiperparámetros se determinaron por ensayo y error.
- Para el caso de *GMM* tuvimos inconvenientes con el tiempo de ejecución que demoraríamos por imagen, esto debido al procesamiento de matrices que realiza internamente el algoritmo. Si bien se podía aplicar el algoritmo para un tamaño menor de imagen ( $20 \times 20$ ) en un tiempo razonable, esta perdía demasiada información y no segmentaría correctamente. Se deja un ejemplo de lo que se obtendría si redujéramos el tamaño, o las iteraciones.
- Para reducir la dimensionalidad de la estructura de datos que utilizaremos para las consultas, se utilizó un algoritmo propio, dado que los rangos entre la cantidad de píxeles de cada conjunto de datos eran muy variados. Este algoritmo duplica ciertos puntos de un conjunto de datos y remueve otros, de forma que se llega a un tamaño único para cada conjunto de datos y, de esa forma, se puede usar apropiadamente la estructura de datos.

- La estructura de datos sobre la cual realizaremos las consultas, una vez procesados los algoritmos de *clustering*, es un ***BallTree***. Este tipo de árbol (que entra en la categoría de "árboles métricos") permite que los *clusters* puedan ser no balanceados. Se cumple también que un dato solo puede estar asignado a un *cluster*. Dado que una hipersfera se ajusta más a una distancia radial que un *MBR*, consideramos que esta estructura es más adecuada que un *KDTree*.

#### B. Kmeans

Aplicamos el algoritmo sobre la figura 0.jpg. La figura 4 muestra todas las clases en las que se dividió la imagen original, y la figura 5 muestra solo el *cluster* que representa al tumor.

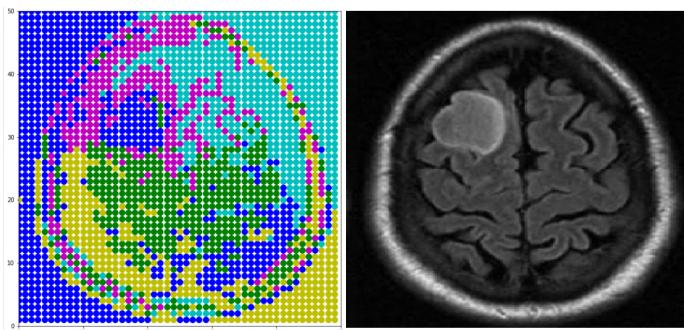


Fig. 4. Aplicación de Kmeans 1

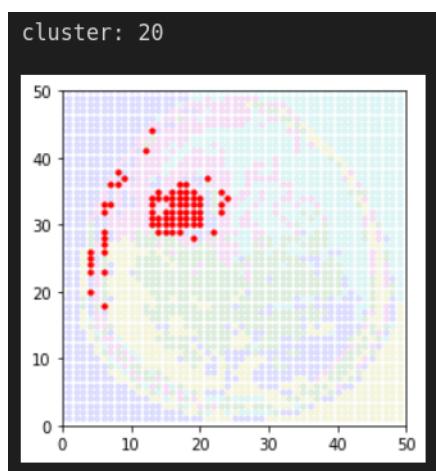


Fig. 5. Aplicación de Kmeans 2

#### C. DBScan

Aplicamos el algoritmo sobre la figura 0.jpg. La figura 6 muestra todas las clases encontradas, mientras que la figura 7 muestra la clase seleccionada.

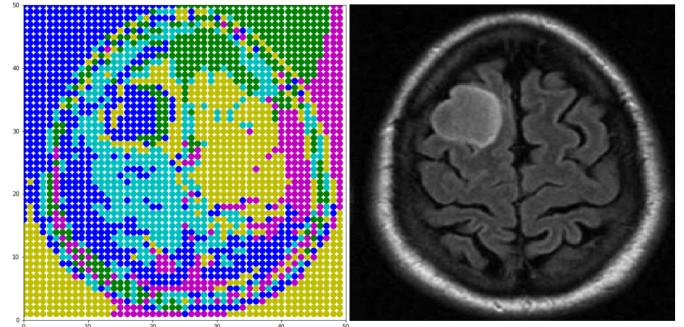


Fig. 6. Aplicación de DBScan 1

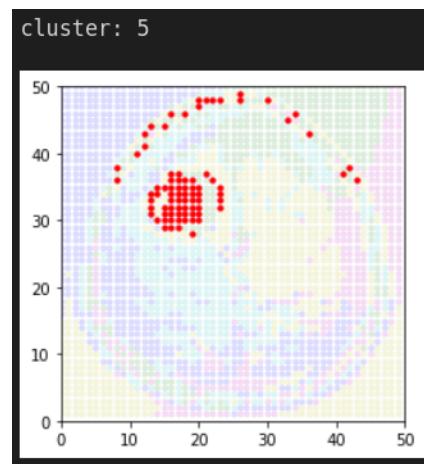


Fig. 7. Aplicación de DBScan 2

#### D. GMM

Aplicamos el algoritmo sobre la figura 0.jpg. En este caso, para un *K* igual a 16, el algoritmo demoró 18 minutos. Y los resultados se muestran en las figuras 8 y 9. En este caso, el valor de *K* debería ser mayor para obtener una mejor segmentación. Por otro lado, si reducimos el tamaño para obtener un tiempo considerable, los resultados no son buenos, y se muestran en las figuras 10 y 11, por lo que se decidió no utilizar este algoritmo para todas las imágenes.

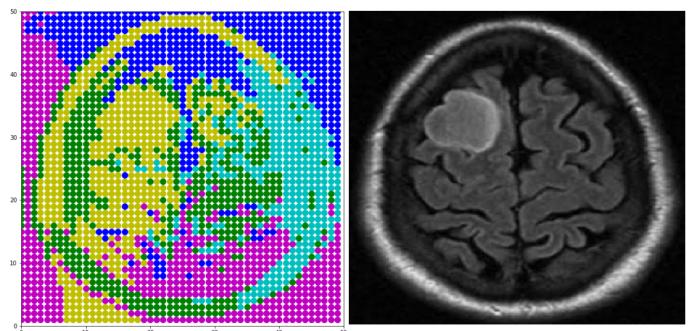


Fig. 8. Aplicación de GMM 1

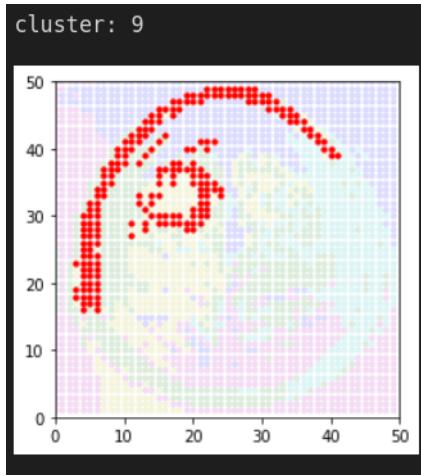


Fig. 9. Aplicación de GMM 2

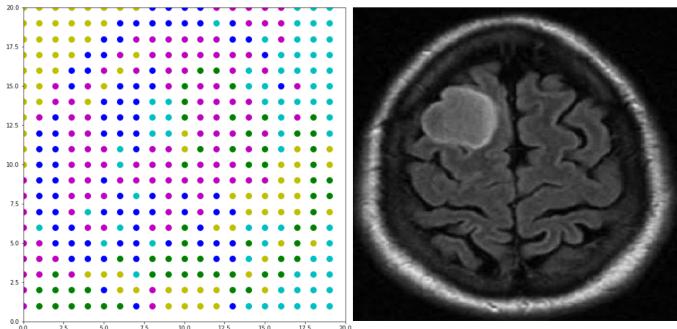


Fig. 10. Aplicación de GMM 3

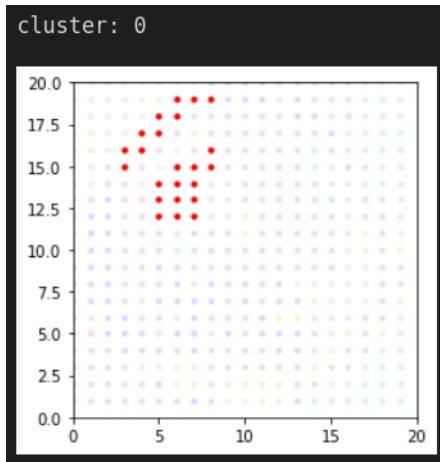


Fig. 11. Aplicación de GMM 4

#### IV. REDUCCIÓN DE DIMENSIONALIDAD E INDEXACIÓN

En esta sección, veremos las técnicas que utilizamos para almacenar y recuperar información relevante a los modelos implementados para este proyecto, de forma que puedan clasificarse nuevas tomografías con el fin de detectar tumores.

##### A. Vector Característico

Para esta implementación, los modelos desarrollados; *Kmeans*, *DBScan* y *GMM*, consideran un vector 3-dimensional que almacenan su posición en los ejes vertical y horizontal y, además, su valor en escala de grises. Considere, entonces, un conjunto de vectores tal como se puede apreciar en la Figura 12, cuya estructura es la antes mencionada.

X	Y	RB
[32, 28, 172]	[33, 28, 198]	[34, 28, 197]
[36, 27, 190]	[35, 26, 204]	[36, 26, 189]
[25, 21, 187]	[33, 21, 177]	[34, 21, 175]
[16, 18, 185]	[30, 18, 191]	[31, 18, 185]
[44, 13, 193]	[43, 12, 198]	[42, 10, 204]

Fig. 12. Estructura de los vectores característicos

Este formato de vectores resulta especialmente práctico, puesto a que se calibra fácilmente a estructuras de datos que utilizaremos para, entre otras cosas, "acercar" las tomografías cuyos rasgos se asemejan entre sí.

##### B. Indexación

La distinción de la clase que contiene el tumor de las otras es un proceso manual y que induce al error humano. Es, también, la etapa donde almacenamos los vectores característicos de aquellos píxeles que se encuentran en dicha clase. Se elaboró un *script* que, dado un *cluster* y un algoritmo, almacena los píxeles de dicho *cluster* en memoria secundaria (siguiendo el formato discutido en la sub-sección anterior). Para los algoritmos de *Kmeans* y *DBScan*, se guardan, específicamente, los siguientes parámetros:

- La cantidad de puntos (píxeles) que se encuentran clasificados como parte del tumor a estudiar.
- Los vectores 3-dimensionales que representan dichos puntos.

A continuación, experimentaremos con la dimensionalidad de estos vectores. Primero, veremos estrategias que nos permiten homogenizar las tomografías; esto es, hacerlas comparables mediante una modificación en la cantidad de píxeles que contiene la clase asignada al tumor. Durante el proceso de identificación de las clases de tumores, se observó que los resultados dependían considerablemente del color del tumor. Esto nos motivó a realizar experimentos donde utilizamos, únicamente, las posiciones de los píxeles de la clase, con el fin de determinar si su ausencia degradaría la precisión de los algoritmos.

##### C. Homogenización

Si un vector característico tiene tamaño  $n$ , y deseamos llevarlo a un tamaño  $x > n$ , duplicamos el contenido del vector hasta superar  $x$ , y finalmente escogemos los primeros  $x$  valores.

Si un vector característico tiene tamaño  $n$ , y deseamos reducirlo a un tamaño  $x < n$ , escogemos los  $x$  datos más cercanos a la media de los datos con ayuda de alguna estructura de datos como el *BallTree*.

## V. CONSULTAS

### A. Homogenización por la media de pixeles por imagen

En esta primera parte, homogenizamos todas las imágenes extraídas a la media de pixeles de todas las imágenes. En el archivo *data\_Kmeans.json* obtenemos una media de 67 pixeles por imagen, y en *data\_DBScan.json* una media de 77 pixeles. Una vez homogenizado estos pixeles para todas las imágenes, obtendremos un vector característico de tamaño  $67 \times 3 = 201$  para el primer archivo y  $77 \times 3 = 231$  para el segundo.

1) *Consultas 1 y 2 en data\_Kmeans.json*: Realizamos dos consultas sobre las imágenes *369.jpg* (Figuras 13 y 14) y *383.jpg* (Figuras 15 y 16)

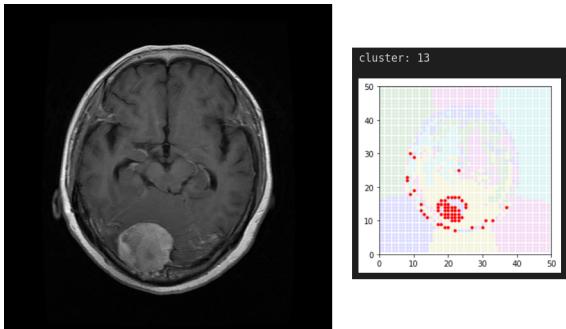


Fig. 13. Consulta 1 sobre *Kmeans*

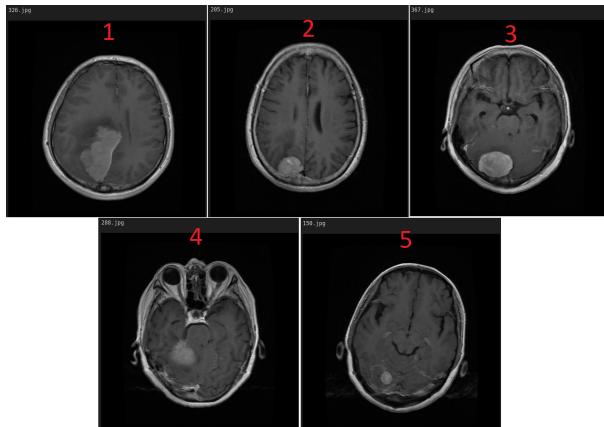


Fig. 14. Resultado sobre la consulta 1 de *Kmeans*

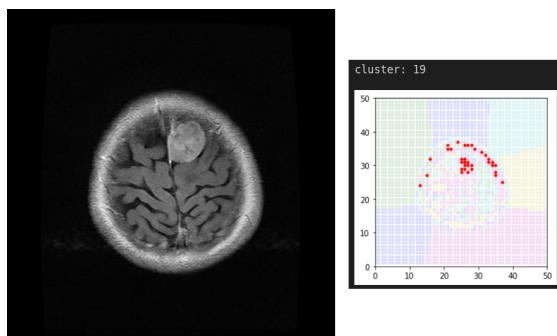


Fig. 15. Consulta 2 sobre *Kmeans*

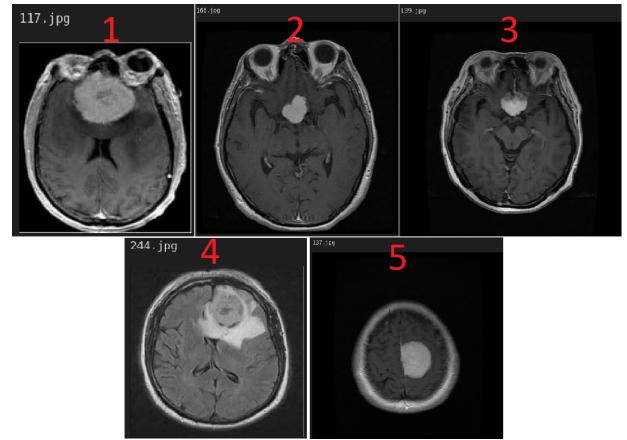


Fig. 16. Resultado sobre la consulta 2 de *Kmeans*

2) *Consultas 1 y 2 en data\_DBScan.json*: Realizamos dos consultas sobre las imágenes *387.jpg* (Figuras 17 y 18) y *398.jpg* (Figuras 19 y 20)

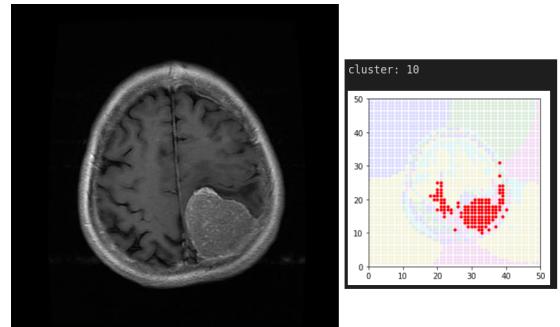


Fig. 17. Consulta 1 sobre *DBScan*

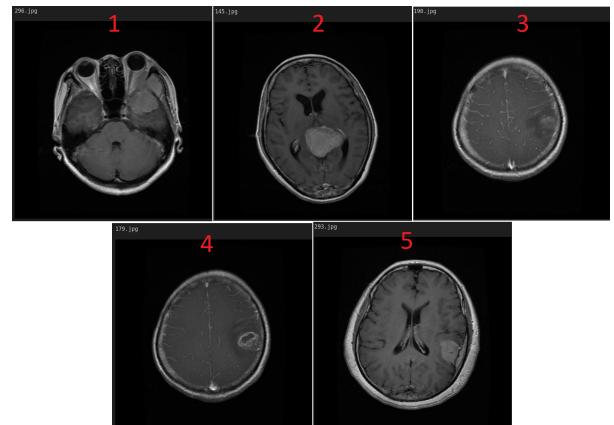


Fig. 18. Resultado sobre la consulta 1 de *DBScan*

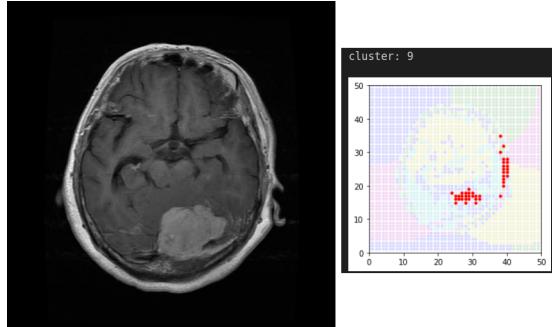


Fig. 19. Consulta 2 sobre *DBScan*

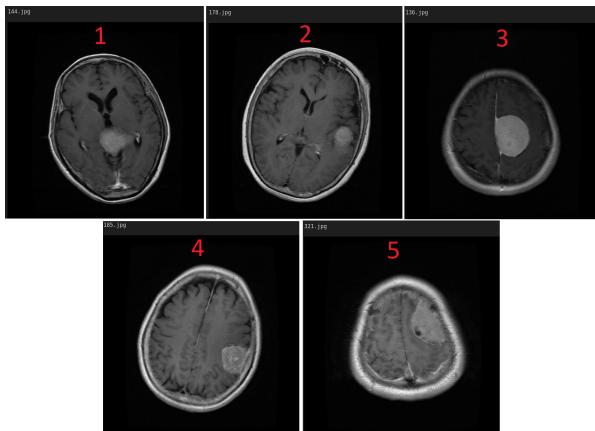


Fig. 20. Resultado sobre la consulta 2 de *DBScan*

Como se observa ambos algoritmos entregan buenos resultados sobre las consultas realizadas. Debemos destacar que los resultados obtenidos están fuertemente influenciados por el color del tumor de la consulta.

#### B. Homogenización por el máximo de pixeles por imagen

En esta segunda parte, homogenizamos todas las imágenes extraídas al máximo de pixeles de todas las imágenes. En el archivo *data\_Kmeans.json* obtenemos un máximo de 312 pixeles por imagen, y en *data\_DBScan.json* un máximo de 441 pixeles. Una vez homogenizado estos pixeles para todas las imágenes, obtendremos un vector característico de tamaño  $312 \times 3 = 936$  para el primer archivo y  $441 \times 3 = 1323$  para el segundo.

1) Consultas 3 y 4 en *data\_Kmeans.json*: Realizamos dos consultas sobre las imágenes *373.jpg* (Figuras 21 y 22) y *375.jpg* (Figuras 23 y 24)

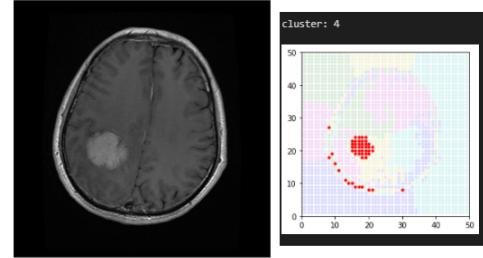


Fig. 21. Consulta 3 sobre *Kmeans*

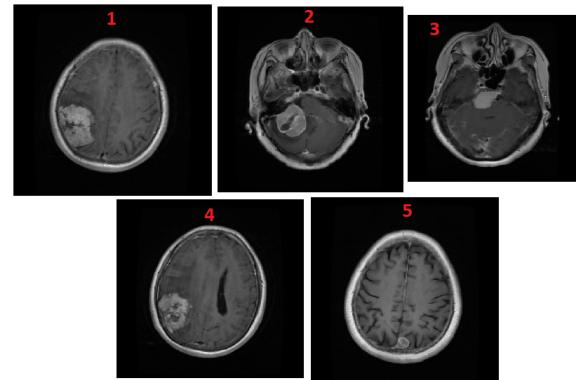


Fig. 22. Resultado sobre la consulta 3 de *Kmeans*

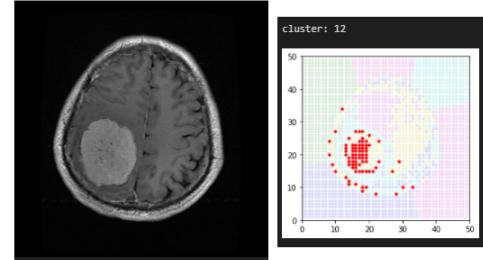


Fig. 23. Consulta 4 sobre *Kmeans*

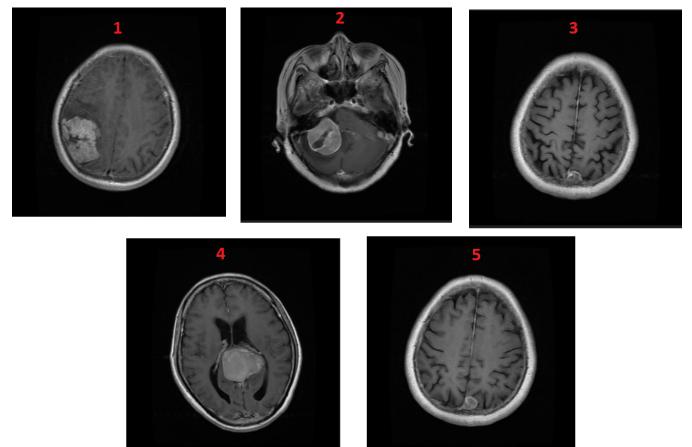


Fig. 24. Resultado sobre la consulta 4 de *Kmeans*

2) Consultas 3 y 4 en *data\_DBScan.json*: Realizamos dos consultas sobre las imágenes 390.jpg (Figuras 25 y 26) y 396.jpg (Figuras 27 y 28)

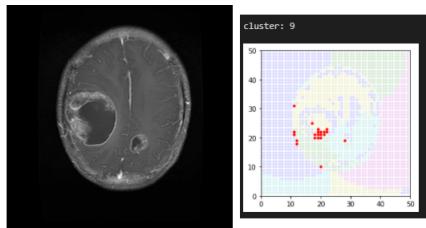


Fig. 25. Consulta 3 sobre *DBScan*

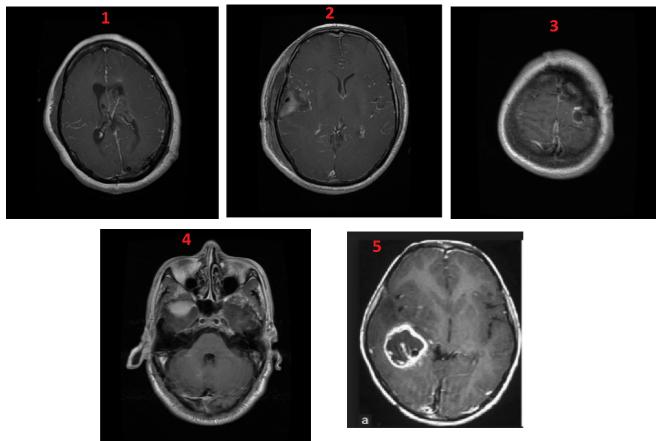


Fig. 26. Resultado sobre la consulta 3 de *DBScan*

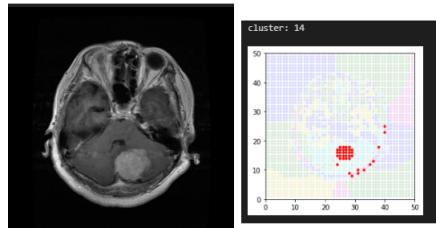


Fig. 27. Consulta 4 sobre *DBScan*

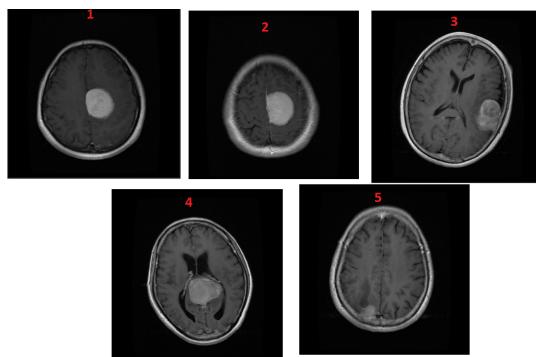


Fig. 28. Resultado sobre la consulta 4 de *DBScan*

Observando las imágenes y las diferentes consultas, podemos encontrar diversos resultados que, a pesar de no ser del todo precisos, se encuentran características en común entre el *input* y los 5 vecinos más cercanos. Podemos observar que todas las imágenes obtenidas tienen un color similar al tumor y están relativamente cerca, en cuanto a posición, al tumor ingresado.

#### C. Homogenización por la media de pixeles por imagen y usando las dos primeras dimensiones

En esta tercera parte, realizaremos el mismo proceso que la primera, solo que esta vez escogeremos las dos primeras dimensiones correspondientes a la posición. En el archivo *data\_Kmeans.json* obtenemos una media de 67 pixeles por imagen, y en *data\_DBScan.json* una media de 77 pixeles. Una vez homogenizado estos pixeles para todas las imágenes, obtendremos un vector característico de tamaño  $67 \times 2 = 134$  para el primer archivo y  $77 \times 2 = 154$  para el segundo.

1) Consultas 5 y 6 en *data\_Kmeans.json*: Realizamos dos consultas sobre las imágenes 357.jpg (Figuras 29 y 30) y 362.jpg (Figuras 31 y 32)

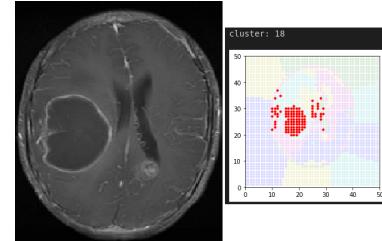


Fig. 29. Consulta 5 sobre *Kmeans*

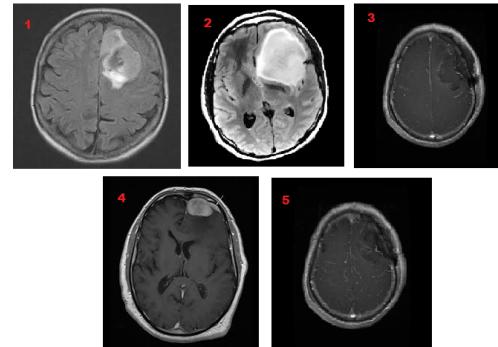


Fig. 30. Resultado sobre la consulta 5 de *Kmeans*

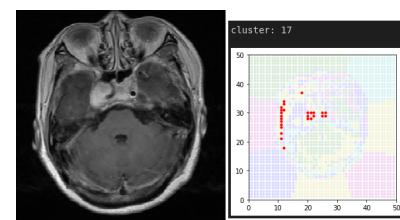


Fig. 31. Consulta 6 sobre *Kmeans*

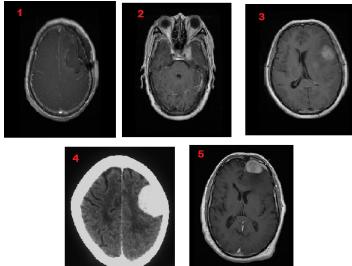


Fig. 32. Resultado sobre la consulta 6 de *Kmeans*

2) Consultas 5 y 6 en *data\_DBScan.json*: Realizamos dos consultas sobre las imágenes 368.jpg (Figuras 33 y 34) y 393.jpg (Figuras 35 y 36)

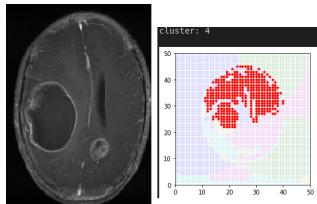


Fig. 33. Consulta 5 sobre *DBScan*

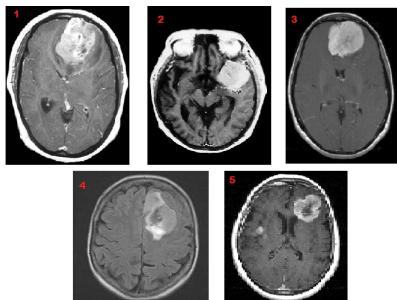


Fig. 34. Resultado sobre la consulta 5 de *DBScan*

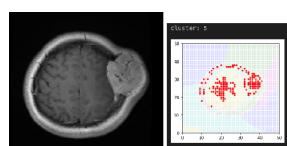


Fig. 35. Consulta 6 sobre *DBScan*

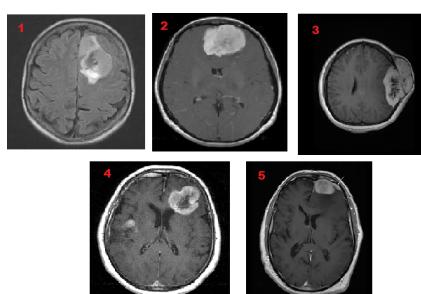


Fig. 36. Resultado sobre la consulta 6 de *DBScan*

Se puede observar que los resultados, si bien son parecidos entre sí, no son tan similares a la imagen de consulta, como sí lo eran en experimentaciones anteriores. Esto es, salvo en la tercera tomografía de la sexta consulta, aplicada con el DBScan, la precisión bajó considerablemente al prescindir de la escala de grises.

## VI. CONCLUSIONES

De acuerdo al trabajo realizado, encontramos las siguientes conclusiones:

- Como son algoritmos no supervisados, depende mucho de la elección de hiperparámetros y del criterio de quien realiza la segmentación.
- Además, se obtendrían mejores resultados si se tuviera un mayor conocimiento en la identificación de tumores cerebrales, ya que en algunas imágenes se nos dificultó reconocer el tumor.
- El modelo *DBScan* resultó ser el más eficiente en cuanto a tiempo, pero más volátil debido a la variabilidad de los hiperparámetros.
- El modelo *GMM* debe ser implementado con librerías que optimicen las operaciones de matrices, ya que esto nos impidió aplicar el algoritmo a todas las imágenes. Incluso sería mejor implementarlo en un lenguaje como C++ que utiliza procesamiento en paralelo.
- Las matrices del modelo *GMM* deben cumplir ciertos requerimientos, de lo contrario el algoritmo no podrá operar. Se implementó un bucle para que ejecute el algoritmo nuevamente en caso encuentre una matriz no válida para las funciones, esto debido al randomizado de las matrices iniciales.
- En la parte de detección de tumores, al ser una tarea manual, se tenía que ser muy cuidadoso puesto que esta parte está sujeta a posibles errores humanos, lo cual puede afectar en la predicción del resultado.
- La precisión de las consultas está fuertemente influenciada por el valor de la escala de grises de los píxeles que la conforman. Pudimos observar que las experimentaciones realizadas con las 3 dimensiones dieron resultados positivos. Sin embargo, ciertamente se percibe una degradación de la calidad de *outputs* al usar únicamente las posiciones *x* e *y* para cada píxel.

## VII. CÓDIGO

### A. Source Github

[https://github.com/jneirar/ia\\_project\\_2](https://github.com/jneirar/ia_project_2)

### B. Dataset de tumores en cerebros humanos

[https://drive.google.com/file/d/1j\\_EDWso-zm5wcPQMpdwKyIg1ZuD7jeBI/view?usp=sharing](https://drive.google.com/file/d/1j_EDWso-zm5wcPQMpdwKyIg1ZuD7jeBI/view?usp=sharing)